

Administración de Bases de Datos

Tema 8. Técnicas de Recuperación en SGBD

Pedro Pablo Alarcón Caveró
Juan Garbajosa Sopeña

*Departamento O.E.I.
Escuela Universitaria de Informática
Universidad Politécnica de Madrid*

Indice

- 8.1. Conceptos de recuperación
- 8.2. Operaciones de lectura/escritura
- 8.3. Transacciones
- 8.4. Diarios para recuperación
- 8.5. Doble paginación
- 8.6. Procedimientos de recuperación

8.1. Conceptos de Recuperación

- ◆ Consiste en devolver a la base de datos a un estado consistente y con la menor pérdida de información y tiempo posible, e incluye:
 - acciones durante el proceso normal de transacciones
 - acciones después de un fallo
- ◆ Los accesos a la BD se realizan a través de transacciones
- ◆ Errores catastróficos
- ◆ Errores no catastróficos

RECUPERAR IMPLICA PODER REPETIR UNO A UNO LOS PROCESOS QUE HAN ACTUALIZADO LA BD
PARA ELLO ES NECESARIO QUE LAS TRANSACCIONES SE EJECUTEN SEGUN UN ESQUEMA DE **SERIALIZACION**

8.1. Tipos de almacenamiento

- ◆ Tipos de almacenamiento
 - **Almacenamiento volátil**: no sobrevive a las caídas del sistema
 - **Almacenamiento no volátil**: disco, cinta...: existen accidentes
 - **Almacenamiento estable frente al no estable**: la información no se pierde “nunca”, se repite en varios medios no volátiles (disco) con modos de fallo independientes
- ◆ La BD reside en almacenamiento no volátil

8.1. Algunas características de los sistemas basados en transacciones

Operación --> Transacción: proceso con **conexión** y **desconexión**

- ◆ *Problema:* Ejecución concurrente de transacciones
- ◆ *Requisito:* Ejecución **correcta** de transacciones



Se estudió en el
tema de Control de
Accesos Concurrentes

- ◆ *Problema:* Incidentes afectan a la ejecución de las transacciones
- ◆ *Requisito:* Tras el incidente la BD debe contener **toda** la información
- ◆ *Requisito:* Tras el incidente la BD contendrá información **consistente**

8.2. Operaciones básicas lectura/escritura

- ◆ La BD reside en almacenamiento no volátil.
- ◆ Bloque o página: unidad de transferencia de datos entre disco y memoria principal.
- ◆ Transferencia de bloques entre el disco y la memoria principal:
 - **input (X)**: transfiere el bloque físico donde se encuentra X a la memoria principal.
 - **output (X)**: transfiere el bloque de registro intermedio (buffer) donde está X al disco, sustituyendo el bloque físico

8.2. Lectura y escritura entre transacciones y BD

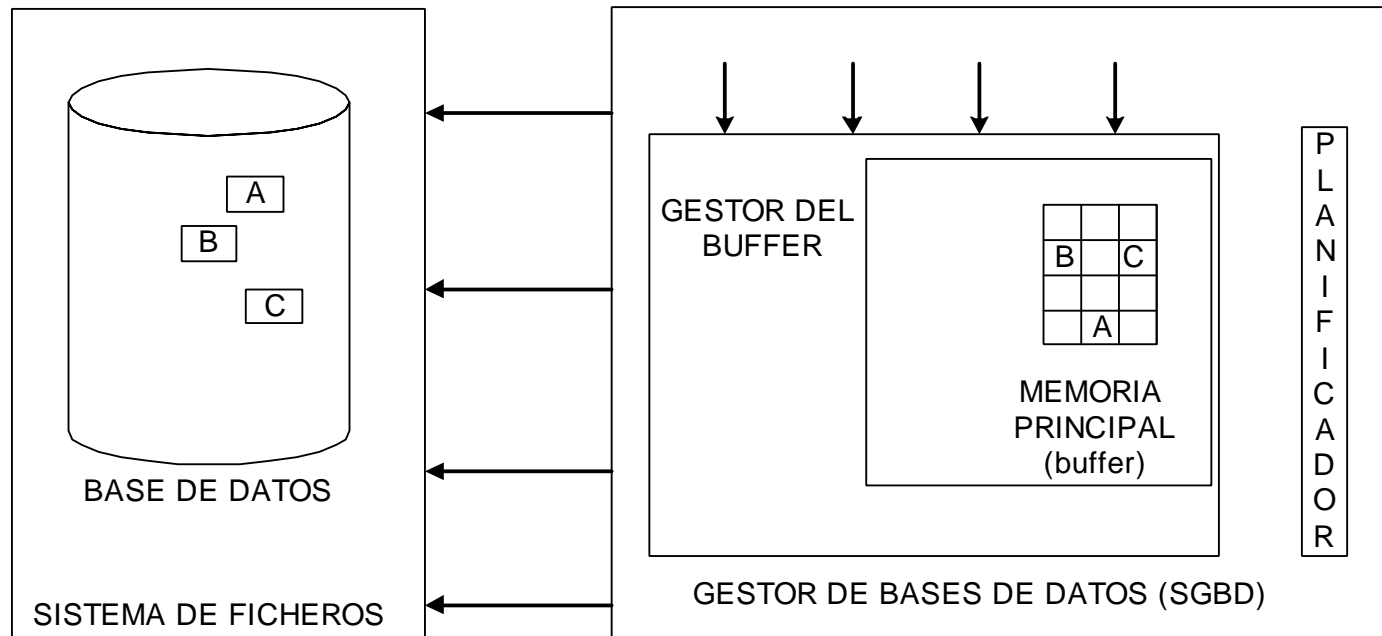
Read (X)

1. Encontrar la dirección donde está X
Bloque o página del disco
2. Copiar el bloque del disco a un buffer de memoria, si no está ya en memoria
3. Copiar X del buffer a la variable X del programa

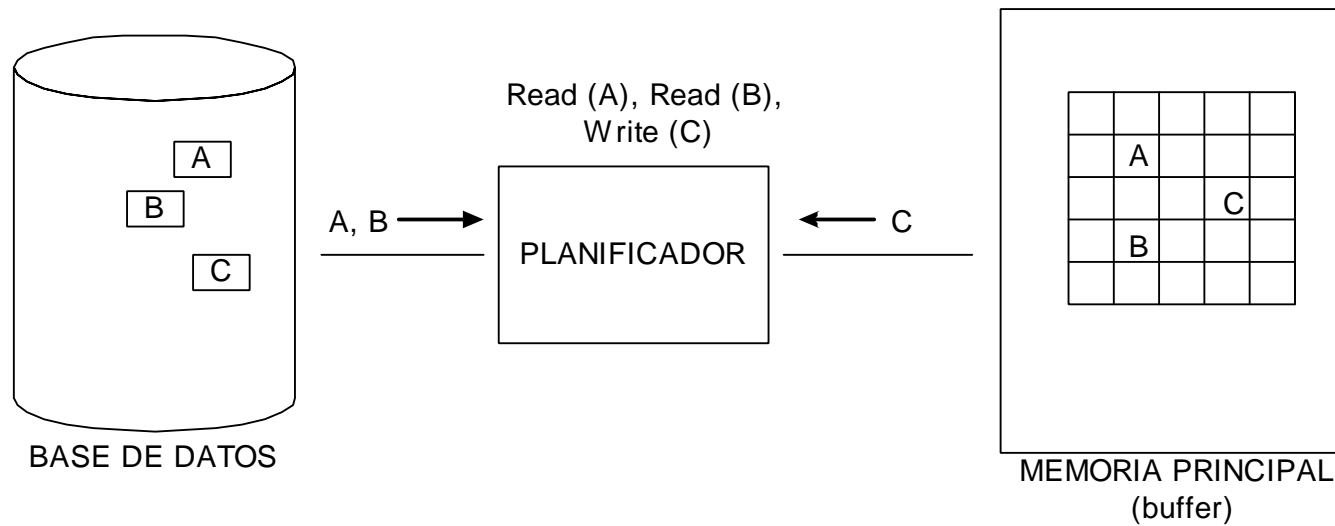
Write (X)

1. Encontrar la dirección donde está X
Bloque o página del disco
2. Copiar el bloque del disco a un buffer de memoria, si no está ya en memoria
3. Copiar X de la variable del programa a su posición en el buffer
4. Almacenar el bloque actualizado desde el buffer al disco

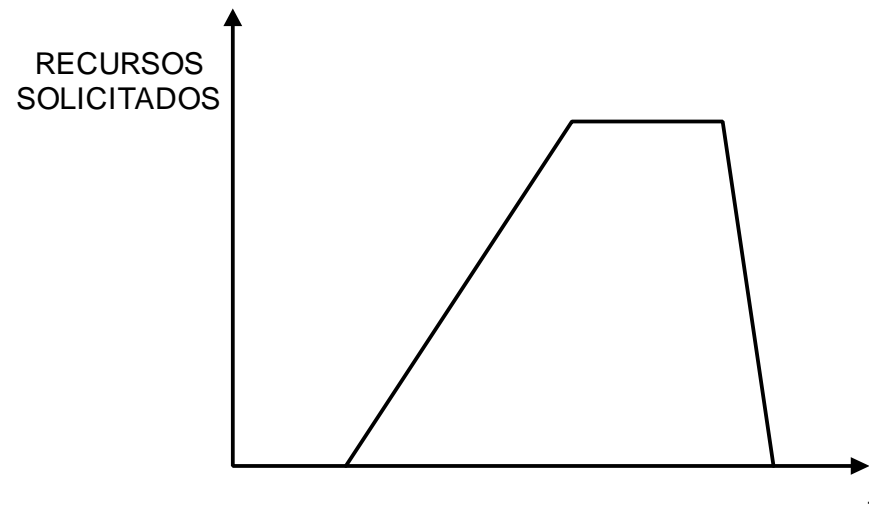
8.2 Funcionamiento del gestor



8.2 Planificador



8.2 Gestión de los recursos



8.2. Gestión en memoria y en disco

- ◆ Un bloque del buffer se graba en disco:
 - porque el gestor de buffer necesita espacio de memoria para otros propósitos
 - o porque el SGBD desea reflejar el cambio hecho a X en el disco
- ◆ Si el sistema se cae tras ejecutar *write* (X, x_i), pero antes de ejecutar *output* (X), el nuevo valor de X se pierde (no se escribe en disco).

8.2. Incidentes que hacen necesaria la recuperación

- ◆ Error lógico

Entrada inválida, información no localizada, etc.

- ◆ Error del sistema

Error de programación, un interbloqueo, etc.

- ◆ Caída del sistema

Se produce una parada del sistema siendo preciso su re-arranque. Se pierde el contenido de la memoria volátil.

- ◆ Fallo de disco

Fallo físico o lógico (de la programación) que implique escrituras incorrectas. Una parte de la memoria secundaria se pierde.

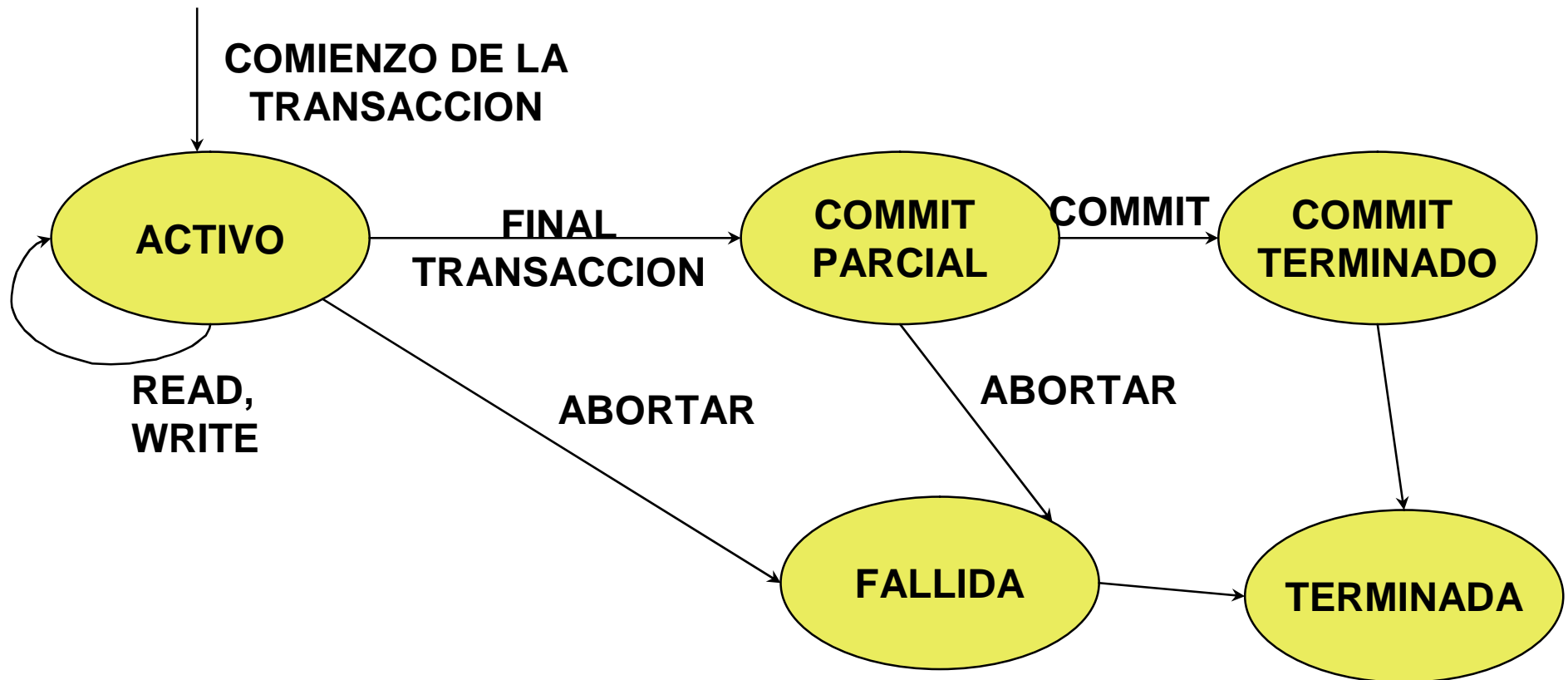
8.2. Efectos de los incidentes

- ◆ La interrupción de la ejecución de una transacción aislada
- ◆ La interrupción de una transacción ejecutándose concurrentemente con otras varias
- ◆ Puede hacer que terminen anormalmente varias
- ◆ Puede destruir físicamente la base de datos, eliminando muchas actualizaciones

Tras la recuperación la **información** de la BD debe estar **completa y consistente**

8.3. Transacciones.

Diagrama de transición de estados



8.3. Operaciones de una transacción

- ◆ READ
- ◆ WRITE
- ◆ BEGIN_TRANSACTION
- ◆ END_TRANSACTION
- ◆ COMMIT_TRANSACTION
- ◆ ROLLBACK
- ◆ ABORT

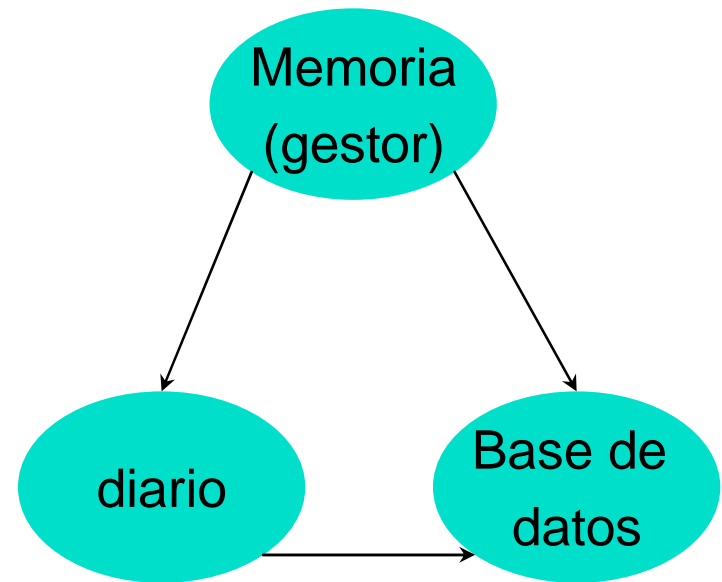
- ◆ UNDO
- ◆ REDO

Necesarias para que se pueda soportar la función de recuperación a través de las transacciones

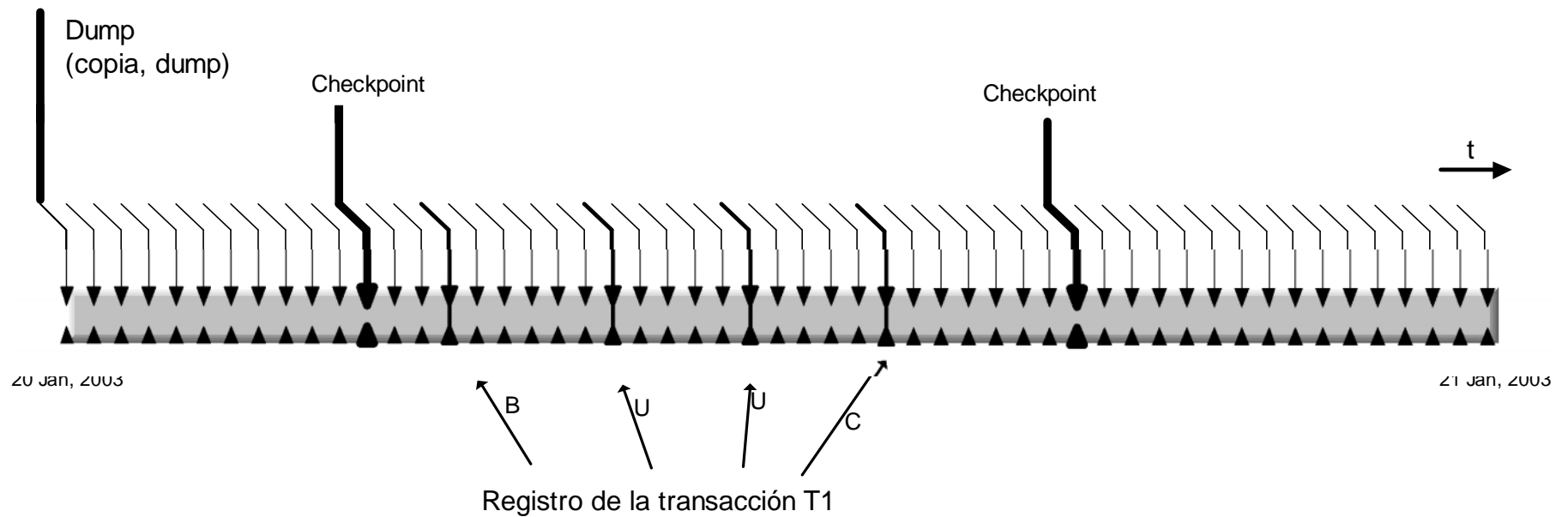
8.4. Diarios para recuperación

- ◆ Se utilizan también los términos *log* y *journal*
- ◆ Mantiene un registro de todas las operaciones que afectan a items de la base de datos
- ◆ Esta información permite recuperar
- ◆ Se almacena en disco
- ◆ Operaciones posibles a reflejar:
 - [start,T]
 - [write,T,X, *valor_viejo*, *valor_nuevo*]
 - [read,T,X]
 - [commit,T]
 - [abort,T]
 - undo, redo,

opcional



8.4 Ejemplo de diario



8.4. Punto de commit de una transacción

- ◆ Se llega al punto de commit cuando
 - todas las operaciones de acceso a la base de datos se han ejecutado con éxito
 - el efecto de las operaciones está almacenado en el diario
- ◆ Cuando se llega al commit los cambios son ya permanentes. Se escribe en el diario [commit,T]
- ◆ En caso de incidente hay que buscar las transacciones que en el diario tengan [comenzar_transacción,T], pero no [commit,T] , y se les hace *roll-back*
- ◆ Llegar al commit point implica que se ha escrito en disco el diario, desde el buffer correspondiente.

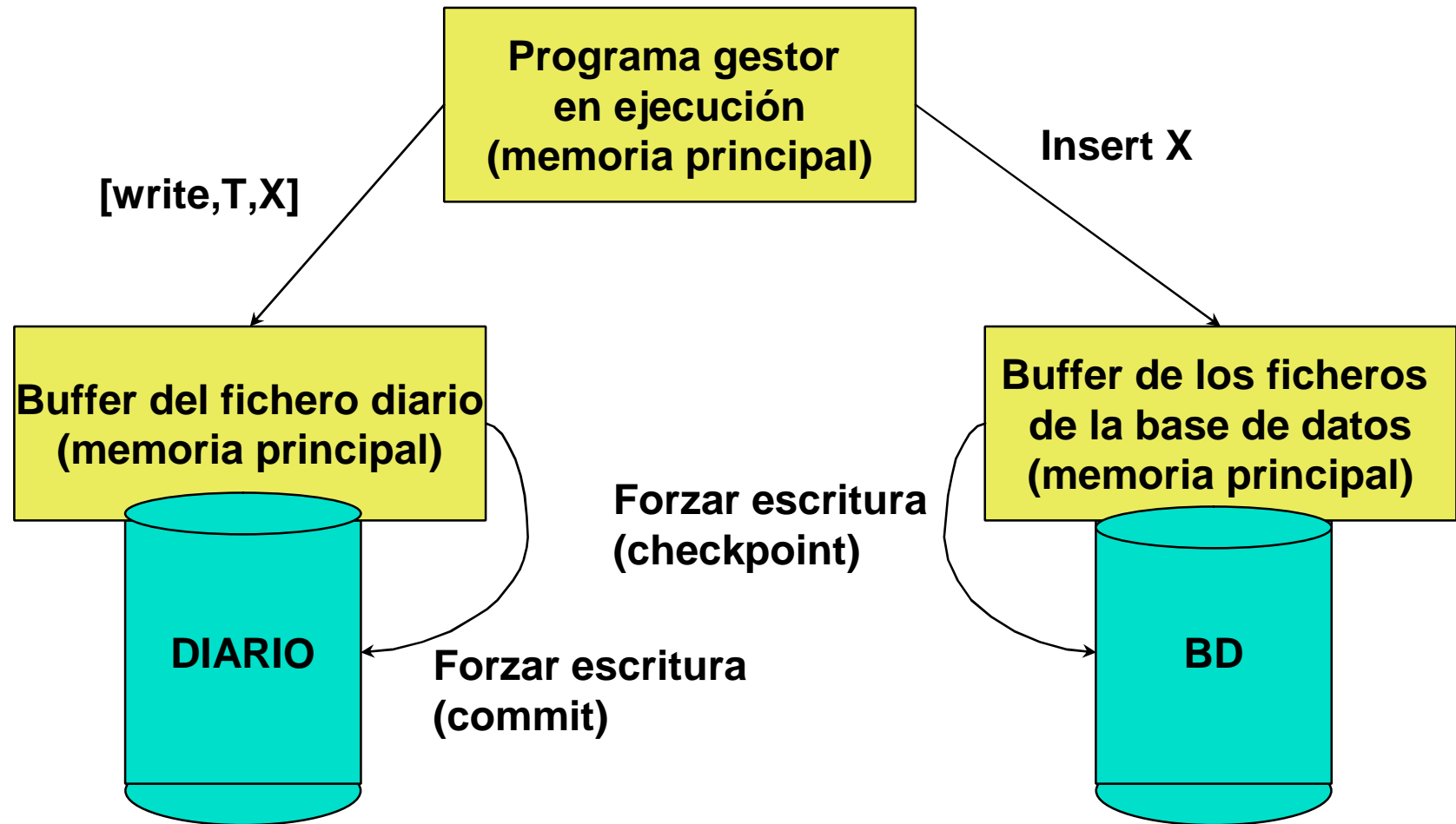
8.4. Tipos de diarios

- ◆ Los diarios recogen la información que permite recuperar
- ◆ Tradicionalmente se distingue:
 - diario con la imagen anterior (Before Image Log):
 - diario con la imagen posterior (After Image Log): la transacción es *committed* cuando el diario ya contiene todos los *write*, aunque no estén en la BD
- ◆ Fichero del sistema (system log) que tiene:
 - valores (imágenes) anteriores y/o posteriores de las páginas modificadas
 - identificadores de las transacciones responsables de las actualizaciones
 - registros que indican el comienzo, validez y anulación de las transacciones

8.4 Checkpoints o puntos de validación

- ◆ Los puntos de *checkpoint* en el diario garantizan que la información de las transacciones que han terminado antes de ese punto está en la base de datos
- ◆ Un checkpoint consistente en:
 - Suspender la ejecución de las transacciones temporalmente
 - Forzar la escritura de todas las actualizaciones de buffer a disco
 - Escribir [checkpoint] en el diario
 - Hacer que las transacciones continúen

8.4. Esquema de ejecución



8.4. Recuperaciones basadas en actualizaciones diferidas (I)

- ◆ Graba todas las actualizaciones de la BD en el diario, pero aplaza la ejecución de todas las operaciones de escritura (write) de una transacción hasta que ésta se encuentre parcialmente cometida.
- ◆ Solamente requiere el nuevo valor del dato.
- ◆ Si la transacción aborta (no llega a *committed*), simplemente hay que ignorar las anotaciones en el diario
- ◆ Para recuperaciones usa el procedimiento:
 - **redo** (Ti), que asigna los nuevos valores a todos los datos que actualiza Ti.

8.4. Recuperaciones basadas en actualizaciones diferidas (II)

- ◆ Después de ocurrir un fallo, se consulta el diario para determinar que transacciones deben repetirse y cuales anularse.
 - Ti debe anularse si el diario contiene el registro *start* pero no el *commit*.
 - Ti debe repetirse si el diario contiene el registro *start* y el *commit*.
- ◆ La operación redo debe ser idempotente, es decir, ejecutarla varias veces debe producir el mismo resultado que ejecutarla una sola vez.

Ejemplo

T_1 : Read (A, a_1)
 $a_1 := a_1 - 5000$
 Write (A, a_1)
 Read (B, b_1)
 $b_1 := b_1 + 5000$
 Write (B, b_1)

T_2 : Read (C, c_1)
 $c_1 := c_1 + 2000$
 Write (C, c_1)

No comparten un mismo gránulo

Diario de actualizaciones diferidas

Diario

<T1 *starts*>
<T1, A, 5000>
<T1, B, 25000>
<T1 *commits*>
<T2 *starts*>
<T2, C, 9000>
<T2 *commits*>

Base de datos

A = 5000; B = 25000

C = 9000

Base de datos puede querer decir memoria o disco!

Recuperación: actualizaciones diferidas

<T1 starts>
<T1, A, 5000>
<T1, B, 25000>

<T1 starts>
<T1, A, 5000>
<T1, B, 25000>
<T1 commits>
<T2 starts>
<T2, C, 9000>

<T1 starts>
<T1, A, 5000>
<T1, B, 25000>
<T1 commits>
<T2 starts>
<T2, C, 9000>
<T1 end>
<T2 commits>

Valores iniciales: A=10000 B=20000 C=7000

1. A=10000 y B=20000, C=7000 (no hay que hacer nada)
2. A=10000?, B= 20000? y C=7000 (redo t1)
3. A=5000?, B=25000? y C=7000 (redo T1, redo T2; después C = 9000)

Los Redo comienzan a hacerse en el último checkpoint!

Interrogación 10000?: no sabemos si la información está en disco o en memoria

8.4. Recuperaciones basadas en actualizaciones inmediatas (I)

- ◆ Permite que las actualizaciones se graben en la BD mientras la transacción está todavía en estado activo (actualizaciones no cometidas).
- ◆ Antes de ejecutar un output (X), deben grabarse en memoria estable los registros del diario correspondientes a X
- ◆ Los registros del diario deben contener tanto el valor antiguo como el nuevo.
- ◆ El esquema de recuperación utiliza dos procedimientos de recuperación:
 - **undo** (Ti): restaura los datos que Ti actualiza a los valores que tenían antes.
 - **redo** (Ti): asigna los nuevos valores a todos los datos que actualiza Ti.

8.4. Recuperaciones basadas en actualizaciones inmediatas (II)

- ◆ Después de ocurrir un fallo, el procedimiento de recuperación consulta el diario para determinar qué transacciones deben repetirse y cuáles deshacerse:
 - Ti debe deshacerse si el diario contiene el registro starts pero no el commit.
 - Ti debe repetirse si el diario contiene el registro starts y el commit.
- ◆ Las operaciones undo y redo deben ser idempotentes para garantizar la consistencia de la BD aun cuando se produzcan fallos durante el proceso de recuperación.

Planteamiento actualizaciones inmediatas

Diario

$\langle T1 \text{ starts} \rangle$

$\langle T1, A, 10000, 5000 \rangle$

$\langle T1, B, 20000, 25000 \rangle$

$\langle T1 \text{ commits} \rangle$

$\langle T2 \text{ starts} \rangle$

$\langle T2, C, 7000, 9000 \rangle$

$\langle T2 \text{ commits} \rangle$

Base de datos

$A = 5000$

$B = 25000$

$C = 9000$

Base de datos puede querer decir memoria o disco!

Recuperación: actualizaciones inmediatas

<T1 <i>starts</i> >	<T1 <i>starts</i> >	<T1 <i>starts</i> >
<T1, A, 10000, 5000>	<T1, A, 10000, 5000>	<T1, A, 10000, 5000>
<T1, B, 20000, 25000>	<T1, B, 20000, 25000>	<T1, B, 20000, 25000>
	<T1 <i>commits</i> >	<T1 <i>commits</i> >
	<T2 <i>starts</i> >	<T2 <i>starts</i> >
	<T2, C, 7000, 9000>	<T2, C, 7000, 9000>
		<T2 <i>ends</i> >
		<T2 <i>commits</i> >

- 1) A y B en la BD a 10000? y 20000? (undo T1)
- 2) A=5000?, B= 25000? y C=7000.(redo T1, undo T2)
- 3) A=5000?, B=25000? y C=9000? (redo T1, redo T2)

8.4. Recuperación hasta un punto de validación

- ◆ 1. Examina el diario hacia atrás hasta localizar un registro <checkpoint>.
- ◆ 2. Considera sólo los registros existentes entre este punto y el final del diario.
- ◆ 3. Ejecuta $\text{undo}(T_j)$ para las transacciones que no tengan registro < T_j commits>, partiendo del final del fichero.
- ◆ 4. Ejecuta $\text{redo}(T_i)$ para las transacciones que tengan su registro < T_i commits>, partiendo desde el punto de verificación hasta el final del diario.

8.4. Criterios de ABD para utilización de diarios

- ◆ La utilización de un tipo u otro de diario depende de la instalación
- ◆ Factores
 - Número de recuperaciones
 - Nivel de concurrencia y granularidad
- ◆ Si pocas transacciones abortan las imágenes antes son adecuadas
- ◆ Si abortan muchas transacciones las imágenes después resultan más eficientes
- ◆ Si hay muchas actualizaciones sobre los mismos gránulos la actualización inmediata es menos eficiente, y las imágenes después pueden ir mejor

8.4. Retrocesos en cascada

- ◆ El fallo en una transacción puede suponer retrocesos en cascada de transacciones que hayan leído datos escritos por una transacción que falló.
- ◆ No son deseables, ya que llevan a deshacer una importante cantidad de trabajo.
 - Se pueden evitar, bajo el bloqueo en dos fases.
 - Los algoritmos de ordenamiento inicial se pueden modificar para evitar el retroceso, pero se introducen esperas (aunque no bloqueos).

8.5. Técnicas de doble paginación

- ◆ Alternativa a las técnicas de recuperación de caídas basadas en diarios.
- ◆ El sistema mantiene dos tablas de paginación durante la vida de una transacción, y son idénticas al comenzar la transacción.
 - **Tabla de paginación actual**
 - » Puede variar cuando la transacción realiza una operación write. Todas las operaciones input y output utilizan esta tabla para localizar las páginas de la BD. Puede almacenarse en memoria volátil.
 - **Tabla de paginación doble**
 - » No se modifica, y debe almacenarse en memoria no volátil.

8.5. Commitment con doble paginación

- ◆ 1. Comprobar que todas las páginas del buffer que haya modificado la transacción se graban en disco.
- ◆ 2. Grabar en disco la tabla de paginación actual.
- ◆ 3. Grabar la dirección en disco de la tabla de paginación actual en la posición fija de memoria estable que contenga la dirección de la tabla de paginación doble. Por tanto, la tabla de paginación actual se convierte en la tabla de paginación doble y la transacción está cometida.

8.5. Ventajas y desventajas de doble paginación

- ◆ Ventajas frente a los diarios :
 - No es necesario aplicar ningún procedimiento de recuperación
 - Se elimina el tiempo para grabar registros.
 - La recuperación de las caídas es más rápida.
- ◆ Desventajas:
 - Fragmentación de los datos.
 - Recolección de basura.
 - La doble paginación es más difícil de adaptar que un diario a los sistemas que permiten ejecución concurrente de transacciones.

8.6. Procedimientos de recuperación

- ◆ Recuperación normal
- ◆ Recuperación en caliente
- ◆ Recuperación en frío

8.6. Recuperación normal

- ◆ Tiene lugar después de una parada normal de la máquina, en la que se escribe un punto de verificación como último registro del diario.
- ◆ Este procedimiento se ejecuta cuando el último registro del diario es un punto de verificación o recuperación del sistema.
- ◆ Este tipo de recuperación también tiene lugar cuando aborta una transacción, debido a la razón que sea.

8.6. Recuperación en caliente

- ◆ Después de un error del sistema
- ◆ Se ejecuta cuando el último registro del diario no es un punto de verificación y el operador no indica pérdida de memoria secundaria.
- ◆ El procedimiento de recuperación es el indicado en el apartado referente a los puntos de verificación en el diario.

8.6. Recuperación en frío

- ◆ Después de un incidente con la memoria masiva dañada
- ◆ Se ejecuta si se pierden datos o la BD ya no es coherente.
- ◆ Se utiliza
 - copia de seguridad (backup) más reciente de la BD
 - diario de las actividades posteriores
 - se aplican las imágenes posteriores al respaldo
- ◆ Puede encadenar una recuperación en caliente.
- ◆ Se deben realizar copias de seguridad de la BD periódicas
 - Toda la BD debe copiarse en memoria secundaria.
 - El proceso de transacciones debe pararse durante el procedimiento de copia

¡Debe existir!

¡Costoso!